

## תוכנה 1



תרגול מספר 12:  
עוד על טיפוסים מוכללים (גנריים)  
יעל אמסטרדמר, אלכסיי זגלסקי

בית הספר למדעי המחשב  
אוניברסיטת תל אביב

1

## שירותים גנריים

- כמו מחלקות גם מתודות יכולות להיות גנריות
- נגדיר את הטיפוס הגנרי לפני טיפוס הערך המוחזר

```
public class Sets {  
    public static <T> Set<T> union(Set<T> s1, Set<T> s2) {  
        Set<T> result = new HashSet<T>(s1);  
        result.addAll(s2);  
        return result;  
    }  
}
```

2

## קוד הלקוח

- בקוד הלקוח לא נצטרך לציין מהו הטיפוס הגנרי
- הקומפילר "יבין" זאת בעצמו מתוך ההקשר

```
public static void main(String[] args) {  
    Set<String> s1 = ...  
    Set<String> s2 = ...  
    Sets.union(s1, s2);  
}
```

3

## הפונקציה max

- כיצד נממש פונקציה המוצאת את האיבר המקסימלי באוסף כלשהו?
- נדרוש יחס סדר על טיפוס האברים

```
public static <T extends Comparable<T>> T max(Collection<T> coll) {  
    if (coll.isEmpty())  
        return null;  
    Iterator<T> i = coll.iterator();  
    T result = i.next();  
    while (i.hasNext()) {  
        T t = i.next();  
        if (t.compareTo(result) > 0)  
            result = t;  
    }  
    return result;  
}
```

עוד נחזור ל-max

4

## תזכורת

- אם Sub הוא תת-טיפוס של Super אז Sub[] הוא תת-טיפוס של Super[]

✓ Sub[] sub = ...  
Super[] sup = sub;

- זה לא נכון לגבי טיפוסים גנריים!  
■ אם T1 ו T2 טיפוסים שונים אז, לדוגמה, בין הטיפוסים <List<T1> ו <List<T2> לא מתקיים יחס של תתי-טיפוסים גם אם יחס כזה מתקיים בין T1 ו T2

✗ List<Sub> sub = new ArrayList<sub>();  
List<Super> sup = sub;

5

## מחסנית

- נתונה המחלקה:

```
public class Stack<E> {  
    public Stack() {...}  
    public void push(E e) {...}  
    public E pop() {...}  
    public boolean isEmpty() {...}  
}
```

- נרצה להוסיף

```
public void pushAll(Collection<E> src) {  
    for (E e : src)  
        push(e);  
}
```

- מה הבעיה במימוש?

6

## הבעיה

מה קורה עבור הקוד הבא:  
(זיכור, Integer יורש מ Number)

```
Stack<Number> numberStack = new Stack<Number>();  
Collection<Integer> integers = ...  
numberStack.pushAll(integers);
```

הודעת שגיאה

```
The method pushAll(Collection<Number>) in the type Stack<Number>  
is not applicable for the arguments (Collection<Integer>)
```

ממה נובעת הודעת השגיאה?

7

## פתרון - Wildcards

שלושה סוגים של wildcards:

1. ?
2. **? extends T**  
קבוצת "כל הטיפוסים" או "טיפוס כלשהו"  
משפחת תתי הטיפוס של T (כולל T)
3. **? super T**  
משפחת טיפוס העל של T (כולל T)

## ? extends E

טיפוס הקלט ל pushAll

במקום "Collection of E" נרצה  
"Collection of **some subtype of E**"

```
public class Stack<E> {  
    ...  
    public void pushAll(Collection<? extends E> src) {  
        for (E e : src)  
            push(e);  
    }  
}
```

חסם עלייון על טיפוס הקלט

E הוא תת טיפוס של עצמו

9

## popAll

כעת נרצה להוסיף את popAll

```
public class Stack<E> {  
    ...  
    public void popAll(Collection<E> dst) {  
        while (!isEmpty())  
            dst.add(pop());  
    }  
}
```

בעיית קומפילציה?

מה עם קוד הלקוח?

## קוד הלקוח

האם יש בעיה בקוד הלקוח?

✓ `Stack<Number> numberStack = new Stack<Number>();`  
`Object o = numberStack.pop();`

✗ `Collection<Object> objects = ...`  
`numberStack.popAll(objects);`

האם השימוש ב extend מתאים גם פה?

## ? super E

טיפוס הקלט ל popAll

במקום "Collection of E" נרצה  
"Collection of **some supertype of E**"

```
public class Stack<E> {  
    ...  
    public void popAll(Collection<? super E> dst) {  
        while (!isEmpty())  
            dst.add(pop());  
    }  
}
```

חסם תחתון על טיפוס הקלט

E הוא תת טיפוס של עצמו

12

## get-put principal

### PECS

Producer Extends Consumer Super

■ השתמשו ב **extends** כאשר אתם קוראים נתונים ממבנה, ב **super** כאשר אתם מכניסים נתונים למבנה ואל תשתמשו ב wildcards כאשר אתם עושים את שניהם

- ב pushAll קוראים נתונים מהמשתנה src
- ב popAll מכניסים נתונים למשתנה dst

## בחזרה ל- max

■ נבחן מחדש את max לפי עקרון ה PECS

האם צרכן או ספק?

```
public static <T extends Comparable<T>> T max(
    Collection<T> coll) {
    if (coll.isEmpty())
        return null;
    Iterator<T> i = coll.iterator();
    ...
    return result;
}
```

האם צרכן או ספק?

14

## בחזרה ל- max

■ נבחן מחדש את max לפי עקרון ה PECS

צרכן

```
public static <T extends Comparable<? super T>> T max(
    Collection<? extends T> coll) {
    if (coll.isEmpty())
        return null;
    Iterator<T> i = coll.iterator();
    ...
    return result;
}
```

ספק

15

## בחזרה ל- max

■ עוד שינוי אחד דרוש

■ התאמת האיטרטור לטיפוס האברים שבאוסף

```
public static <T extends Comparable<? super T>> T max(
    Collection<? extends T> coll) {
    if (coll.isEmpty())
        return null;
    Iterator<? extends T> i = coll.iterator();
    ...
    return result;
}
```

16

## max – מימוש מעט שונה

■ נשתמש בלולאת foreach

```
public static <T extends Comparable<? super T>> T max(
    Collection<? extends T> coll) {
    if (coll.isEmpty())
        return null;
    T candidate = coll.iterator().next();
    for (T element : coll) {
        if (element.compareTo(candidate) > 0)
            candidate = element;
    }
    return candidate;
}
```

17

## Unbounded Wildcard

■ כשלא יודעים או לא אכפת לנו מהו הטיפוס האמיתי

■ לדוגמא, פונקציות הפועלות על מבנה ה collection (shuffle, rotate, ...)

```
static int numberOfElementsInCommon(Set<?> s1, Set<?> s2) {
    int result = 0;
    for (Object o : s1) {
        if (s2.contains(o))
            result++;
    }
    return result;
}
```

18

## שימוש ב ? הוא בטוח

- ניתן להוסיף כל אובייקט ל raw collection לא בטוח!
- לא ניתן להוסיף אובייקטים ל `collection<?>`, בכלל!
  - חוץ מ null
  - שגיאת קומפילציה

19

## שונות

## מנשקים וירושה

```
class A {
    public void print() {
        System.out.println("A");
    }
}

class B extends A implements C {
}

interface C {
    void print();
}
```

public by default

Is there an error?

21

## מנשקים וירושה

```
class A {
    void print() {
        System.out.println("A");
    }
}

class B extends A implements C {
}

interface C {
    void print();
}
```

Is there an error?

22

## דריסת שירותים

```
public class A {
    public void print() {
        System.out.println("A");
    }
}

public class B extends A {
    public void print() {
        System.out.println("B");
    }
}
```

```
public class C {
    public static void main(String args[]) {
        B b = new B();
        A a = b;
        b.print();
        a.print();
    }
}
```

Casting is  
unneded

Does the code compile? If no, why?  
Does the code throw a runtime  
exception?  
If yes, why? If no, what is the output?

## דריסת שירותים

```
public class A {
    public void print() {
        System.out.println("A");
    }
}

public class B extends A {
    protected void print() {
        System.out.println("B");
    }
}
```

```
public class C {
    public static void main(String[] args) {
        B b = new B();
        b.print();
    }
}
```

Does the code compile? If no, why?  
Does the code throw a runtime  
exception?  
If yes, why? If no, what is the output?

24

## בחינה באופק

- הבחינה ב- 9.7
- כל הנושאים שכיסינו במהלך הסמסטר (שיעורים, תרגולים ועבודות בית)
- Java, DBC, ירושה ופולימורפיזם, IO, iterator, ... , Collection Framework, Generics, מחלקות פנימיות, ...
- לפתור כמה שיותר מבחנים משנים שעברו

**בהצלחה!**